

Replication-based Highly Available Metadata Management for Cluster File Systems

Zhuan Chen, **Jin Xiong**, Dan Meng



Institute of Computing Technology
Chinese Academy of Sciences

Cluster 2010, Heraklion, Greece
September 23, 2010

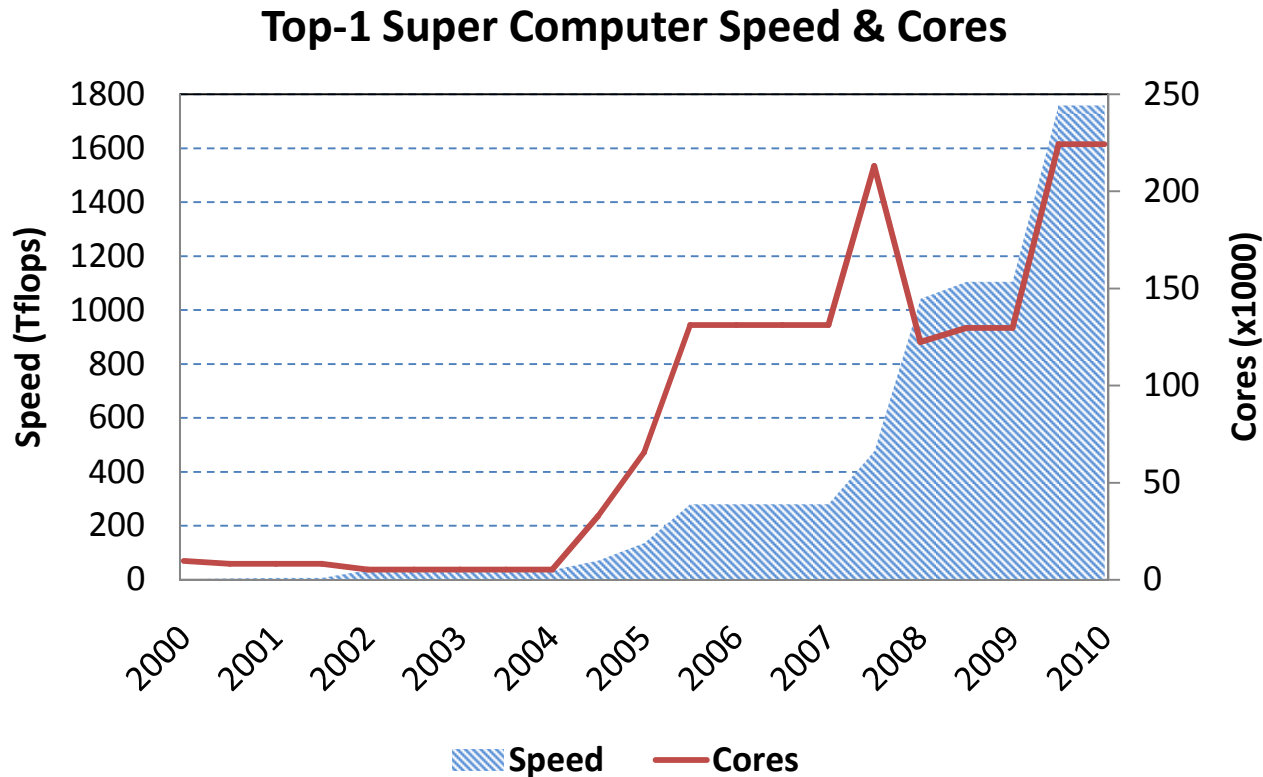
Outline

- ▶ Motivation
- ▶ Related work
- ▶ Our approach
- ▶ Experiment Results
- ▶ Conclusion

Motivation

Speed of supercomputers

- ▶ Speed of supercomputers keeps increasing
 - ▶ 62x from June 2007 to June 2010, according to top500 lists



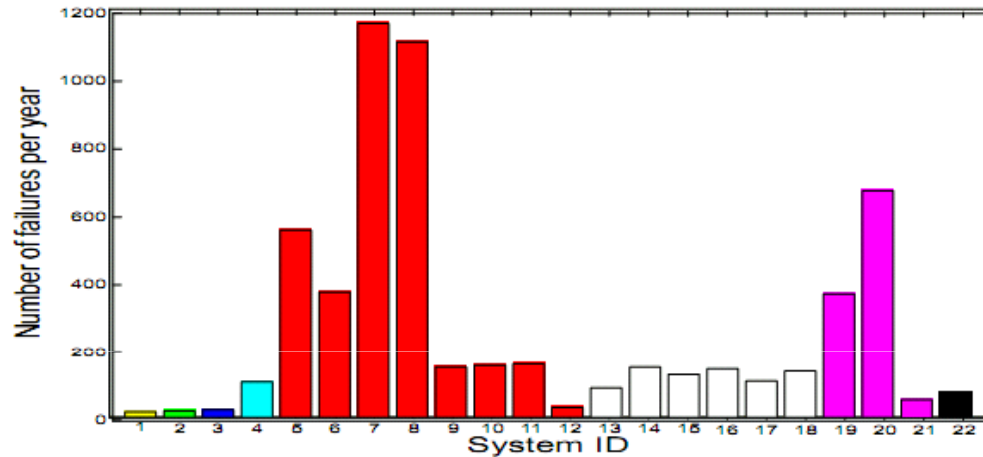
Scale of supercomputers

- ▶ Scale of supercomputers is also increasing
 - ▶ 1,000s to 10,000s nodes
 - ▶ 100,000s processing cores

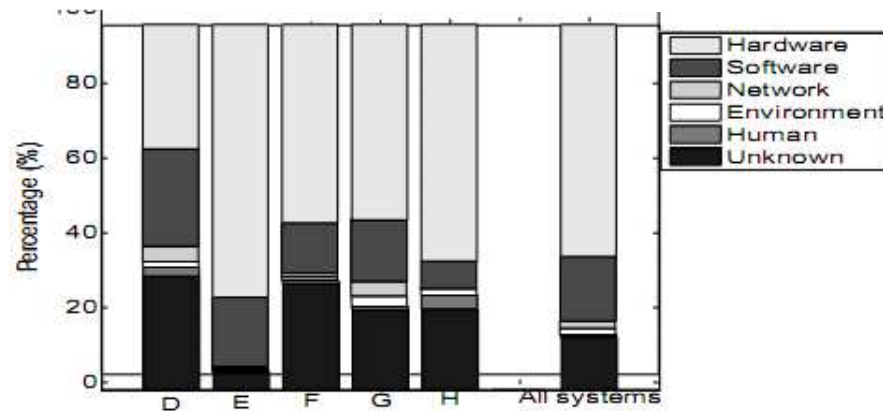
| | Nodes | CPUs | Processing Cores |
|---------------|-----------|--------------------------------|------------------|
| Google, Yahoo | Thousands | | 100s thousands |
| Roadrunner | 3240 | 6,480AMD Opteron 12,960Cell | 129,600 |
| Jaguar | 26520 | 45208 | 255,584 |
| Tera-100 | 4300 | | 140,000 |

Number of failures

- ▶ Component failures are so frequent for supercomputers

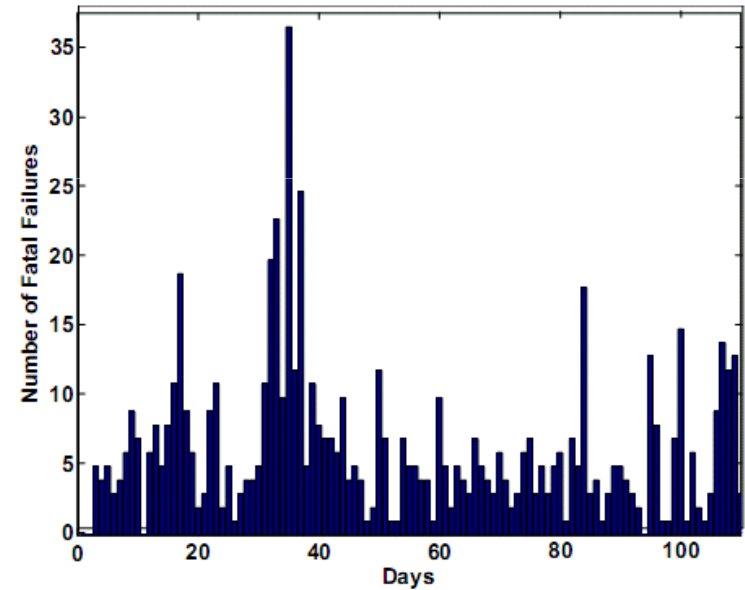


Schroeder & Gibson (DSN'06)



(a)

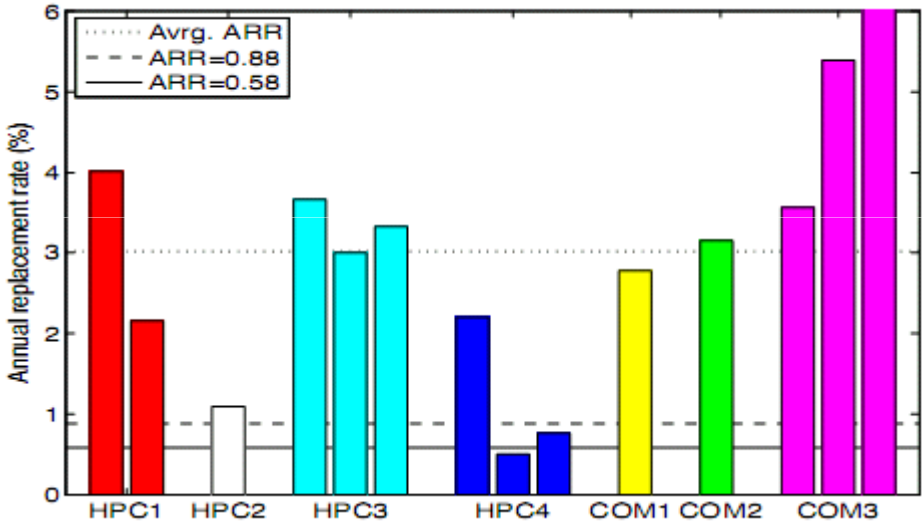
Schroeder & Gibson (DSN'06)



Liang etc. (DSN'06)

Disk failures

► Disk failures are also frequent



Schroeder & Gibson (FAST'07)

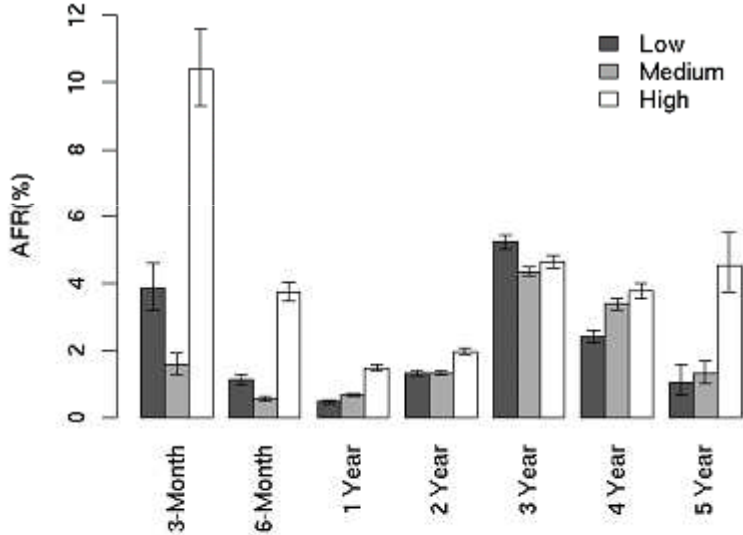


Figure 3: Utilization AFR

Schroeder & Gibson (FAST'07)

File system unavailable

- ▶ Node crashes, disk failures, comm. errors may cause the file system unavailable

| Lustre-FS outage time | | | |
|-----------------------|----------------|----------------|-------|
| Cause of Failure | Start time | End time | Hours |
| I/O hardware | 07/21/07 23:03 | 07/22/07 12:00 | 12.95 |
| I/O hardware | 07/31/07 01:49 | 07/31/07 20:01 | 18.18 |
| I/O hardware | 08/22/07 18:08 | 08/23/07 02:15 | 08.12 |
| I/O hardware | 08/28/07 16:20 | 08/29/07 18:01 | 01.67 |
| I/O hardware | 09/25/07 18:00 | 09/26/07 09:30 | 15.50 |
| I/O hardware | 10/04/07 09:30 | 10/04/07 21:55 | 12.42 |
| Batch system | 10/16/07 17:56 | 10/16/07 21:24 | 03.47 |
| Network | 10/29/07 11:53 | 10/29/07 15:15 | 03.36 |
| File system | 11/16/07 09:30 | 11/16/07 10:00 | 00.40 |
| File system | 11/19/07 09:04 | 11/19/07 11:00 | 01.93 |

Table 1. User notification of outage of the Lustre-FS

| Date | # | Date | # | Date | # |
|----------|-----|----------|-----|----------|-----|
| 07/03/07 | 102 | 07/19/07 | 258 | 08/16/07 | 375 |
| 08/20/07 | 591 | 09/05/07 | 005 | 09/17/07 | 002 |
| 09/18/07 | 004 | 09/19/07 | 003 | 09/28/07 | 463 |
| 09/29/07 | 477 | 10/01/07 | 051 | 10/02/07 | 035 |

Table 2. Lustre mount failure notification by compute nodes from 07/01/07 to 10/02/07; column with “#” represents the number of compute nodes that experienced mount failure

Goankar etc (DSN’08)

Challenge for cluster file system design

- ▶ Handle frequent component failures
 - ▶ Hide the failures from applications
 - ▶ High-performance I/O
- ▶ Data replication is not enough
 - ▶ Only consider failures in data paths
 - ▶ Cluster file systems have separate metadata paths
- ▶ Metadata management
 - ▶ Maintain the namespace, file attributes, data location, access permission, etc
 - ▶ The disruption of metadata service could lead to the outage of the entire file system

Issue to address

- ▶ How to ensure highly available metadata management
 - ▶ Previous work: tolerate failures in the data path through replication

Building Highly Available Cluster File System Based on Replication, L. Cao, Y. Wang, J. Xiong, PDCAT'09, Dec. 2009

- ▶ This work: tolerate failures in the metadata path through replication

Related work

Metadata journaling

- ▶ Adopted by many cluster file systems
- ▶ Ensure atomicity of metadata operations
- ▶ Keep metadata consistency after metadata server failures
- ▶ 2 methods for writing log back to disk
 - ▶ Synchronous write: poor metadata performance by a large number of small and synchronous disk writes
 - ▶ Asynchronous write: better throughput, but operation latency is hardly improved, and loss of memory metadata and state information

Limitation: cannot provide seamless recovery and fail-over

Replication

- ▶ The symmetric active/active metadata service
 - ▶ Ou et al. , PDCS 2007
 - ▶ Total order for all metadata write requests by broadcasting
 - ▶ Large amount of network transmissions
- ▶ Google FS replicates its master on multiple machines
 - ▶ Operation log and checkpoints
 - ▶ Shadow masters (not mirrors) provide read-only access to some files in some situation
- ▶ ...

Challenges

- Reduce costs brought by replication & consensus protocol
- Well utilize the redundant MDS

of applications on supercomputers

Our approach

Background

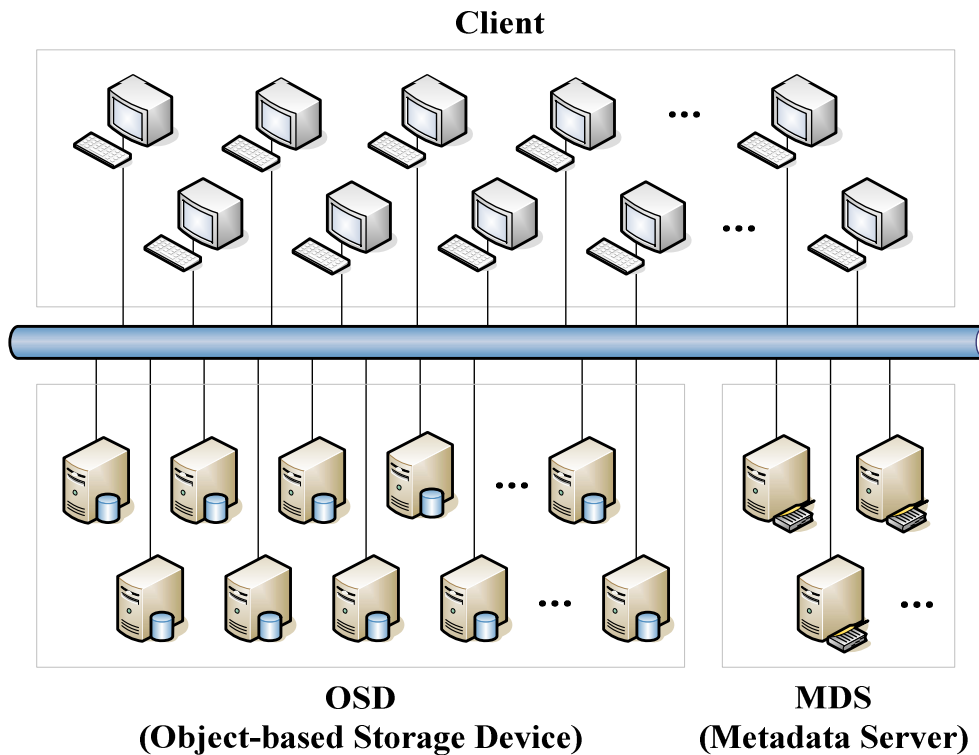
- ▶ Dawning5000A
 - ▶ 1,650 nodes
 - ▶ 30,720 cores
 - ▶ 100TB memory
 - ▶ Linpack: 180.6 Tflops
 - ▶ No.10 at Nov 2008 Top500 list
 - ▶ No. 24 at Jun 2010 Top500 list
 - ▶ Total storage of local disks: >500TB
 - ▶ 320GB SATA disk per node



A high-performance cluster file system over all local disks

Our prototype: DCFS3

- ▶ Global namespace shared by thousands of nodes
- ▶ Logically, 3 types of components
 - ▶ Client, OSD, MDS
- ▶ may co-locate at the same physical node

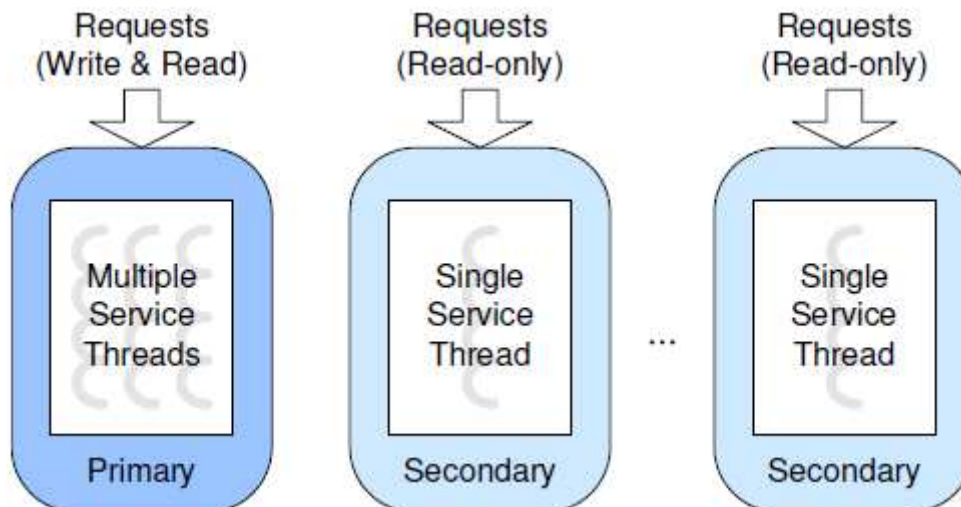


Metadata management in DCFS3

- ▶ Metadata is partitioned according to users
 - ▶ Each partition is also the loading unit
- ▶ Access and updates are performed on memory data
- ▶ Periodically write back dirty data to disk
- ▶ Metadata modification log: metadata consistency
 - ▶ Asynchronously write log back to disk
 - ▶ A pair of log buffers: to avoid blocking of processing

Asymmetric architecture of replicated MDS

- ▶ Each metadata server replicates its data to other MDS
- ▶ Role of MDS in each group: *Primary & secondaries*
- ▶ Clients view
 - ▶ send metadata write operations to the *primary*
 - ▶ send metadata read operations to any MDS
- ▶ *Primary* determines the execution order of write ops
- ▶ *Secondaries* apply updates according to this order



Metadata replication

- ▶ Replicate each write operation's results
- ▶ A log record for each write operation at the *primary*
 - ▶ Operation type, arguments, results , time stamp
 - ▶ Sequence number
- ▶ *Secondaries* directly apply the results from log records
 - ▶ Expedite the replication by saving the time for repeated processing
- ▶ No interactions among the *secondaries* to complete the replication
- ▶ Replication on a *secondary* is completed if the results have been applied to the memory metadata
- ▶ *Primary* replies the client once the replication is completed by all the *secondaries*

Consensus of replicas

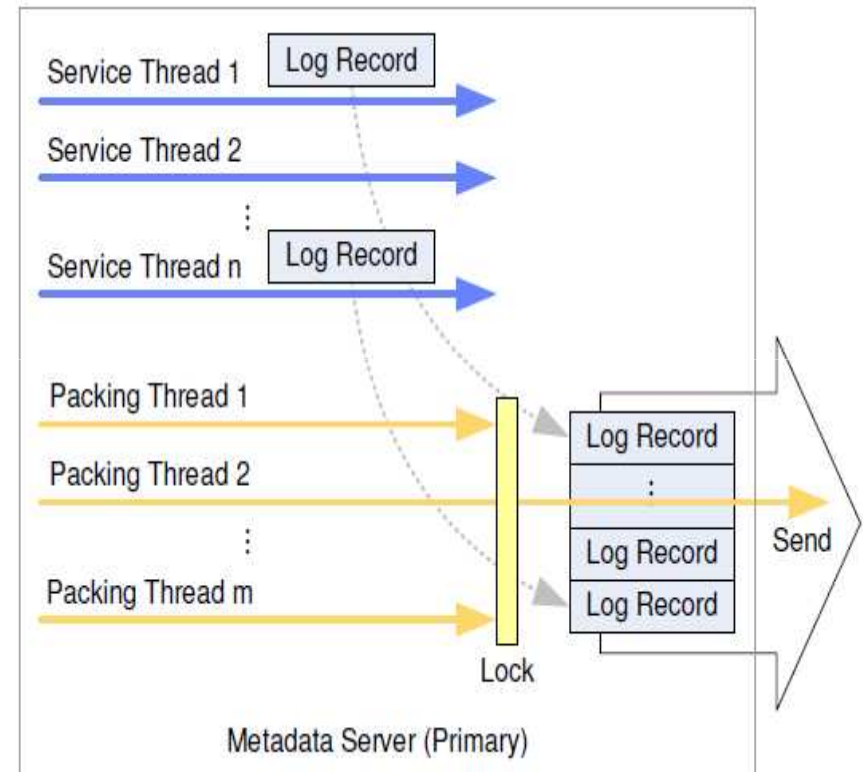
- ▶ Consensus protocol is needed in case of failure occurrence
 - ▶ Paxos algorithm
 - ▶ Primary also plays the role of issuing proposals
 - ▶ 3 rounds of messages: prepare, accept, success
 - ▶ Multi-Paxos
 - ▶ 2 rounds of messages in most cases: accept & success
- ▶ Problem of applying Paxos in metadata management
 - ▶ Cannot make full use of the network bandwidth
 - ▶ A large number of messages under heavy load: every replication requires 2 rounds of network messages
 - ▶ Each message is small: the metadata of each file is small
 - ▶ Poor metadata throughput under heavy load

Packed Multi-Paxos

- ▶ Goal: reduce number of messages in the system
- ▶ Idea: to pack several metadata log records together and transmit them by one message
- ▶ Problem: how many log records to be packed each time?
 - ▶ Trade off between the number of log records per message and the operation latency
 - ▶ Wait for more log records may increase operation latency
 - ☹ Fixed method: unnecessary increase of latency under light load
 - ☹ Fix the number of log records per packing
 - ☹ Fix the time for each packing

Packed Multi-Paxos

- ▶ A self-adaptive method making use of OS thread scheduling
- ▶ Multiple packing threads
 - ▶ Only one active packing thread at any time
 - ▶ The active one packs all current log records into one message
- ▶ Multiple service threads
 - ▶ Do not directly replicate log records to the secondaries
 - ▶ Notify the active packing thread when a log record is generated

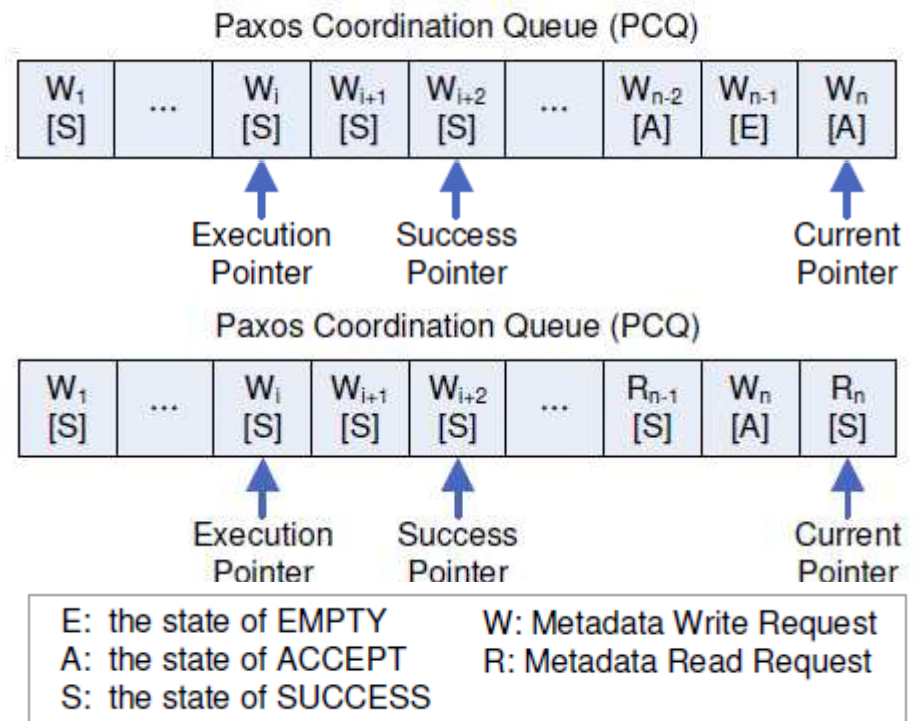


Packed Multi-Paxos

- ▶ Service threads & packing threads are equally scheduled
 - ▶ Same priority
 - ▶ Heavy load: several log records are packed into one message
 - ▶ Several service threads are scheduled before the active packing thread
 - ▶ Improve metadata throughput: less messages and better utilization of network bandwidth
 - ▶ Light load: immediately send log records
 - ▶ Some sleeping service threads (have no task) are just bypassed by the OS scheduler
 - ▶ Immediately execute the active packing thread
 - ▶ Prompt response

Paxos Coordination Queue (PCQ)

- ▶ Control execution order on the secondaries
 - ▶ execution order of log records of write requests
 - ▶ the relevance between metadata write and read requests
 - ▶ 3 states
 - ▶ Accept: log record enqueue
 - ▶ Success: ready to be executed
 - ▶ Receive Success message
 - ▶ Read requests
 - ▶ Empty: discontinuous seqno
- ▶ Execute requests of Success state sequentially
 - ▶ Wait at the request in Accept/Empty state



Recovery

- ▶ Failure of any secondary
 - ▶ Remove it from this group's view
- ▶ Failure of the primary
 - ▶ New primary: largest Accept seqno (or Current Pointer)
 - ▶ Recovery range: smallest Success Pointer to largest Current Pointer
 - ▶ Success State: send this log record to all the Secondaries
 - ▶ Accept State: redo Paxos phase 2, ask if others can accept it
 - ▶ Empty State: ask if others have it, if yes, do as Accept state or Success state; Otherwise, discard it (no-op)

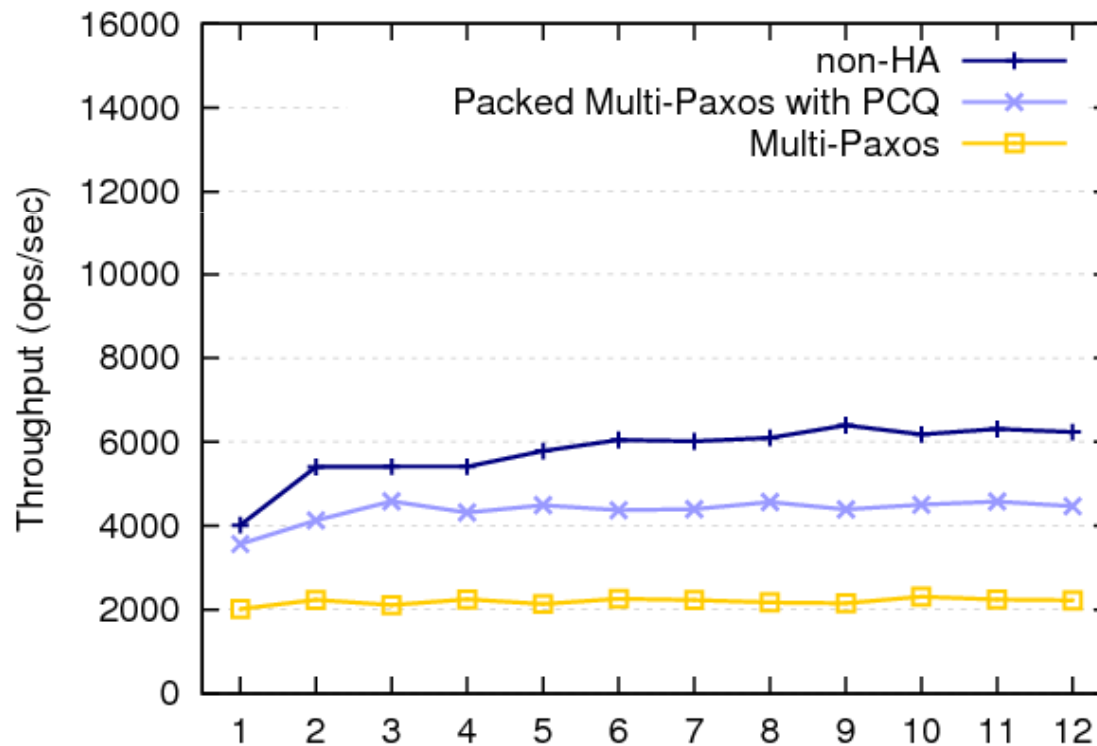
Experiment results

Test platform

- ▶ 16 virtue machines created by VMware
- ▶ Each virtue machine
 - ▶ 1 Intel Xeon Processor (2.00GHz)
 - ▶ 1 GB Memory
 - ▶ 36 GB disk
- ▶ GE interconnection
 - ▶ Actual network transmission speed is 764.91 MB/s (*netperf*)

Performance of metadata write operations

- ▶ Benchmark: *mdtest*
- ▶ Reported performance: file creation



- Configuration:
 - 12 clients, 1 OSD
 - Non-HA: 1 MDS
 - Others: 3 MDS
- *mdtest* parameters
 - 100 threads
 - 100,000 files

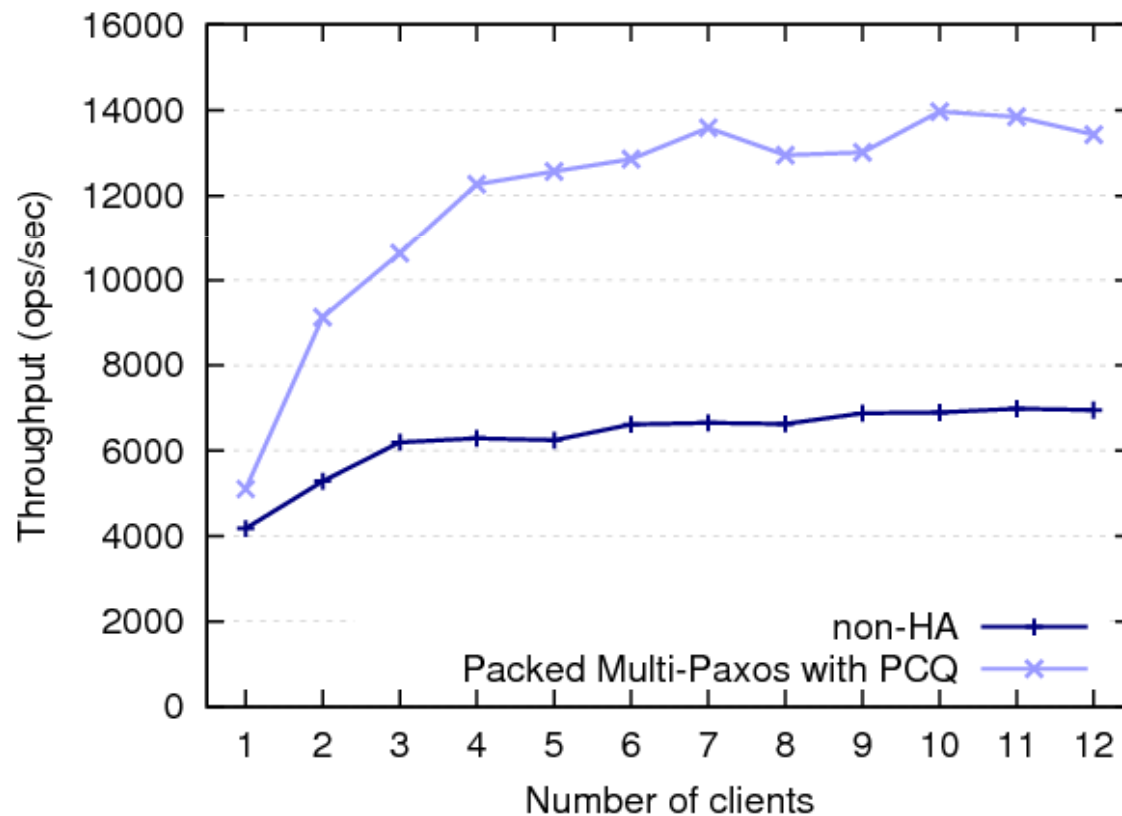
Performance of metadata write operations

- ▶ The effect of packing
- ▶ Reduced 88% network transmissions
- ▶ 90% messages contain 3-6 log records

| Number of log records within a network transmission | Number of network transmission | | |
|---|--------------------------------|-------------------------|--------------------------------------|
| | 1 MDS, non-HA | 3 MDS, HA (Multi-Paxos) | 3 MDS, HA (Packed Multi-Paxos + PCQ) |
| 0 | 0 | 300300 | 304 |
| 1 | 0 | 300300 | 764 |
| 2 | 0 | 0 | 3134 |
| 3 | 0 | 0 | 11136 |
| 4 | 0 | 0 | 20403 |
| 5 | 0 | 0 | 19476 |
| 6 | 0 | 0 | 10148 |
| 7 | 0 | 0 | 2411 |
| 8 | 0 | 0 | 336 |
| 9 | 0 | 0 | 45 |
| 10 | 0 | 0 | 1 |
| Total | 0 | 600600 | 68158 |

Performance of metadata read operations

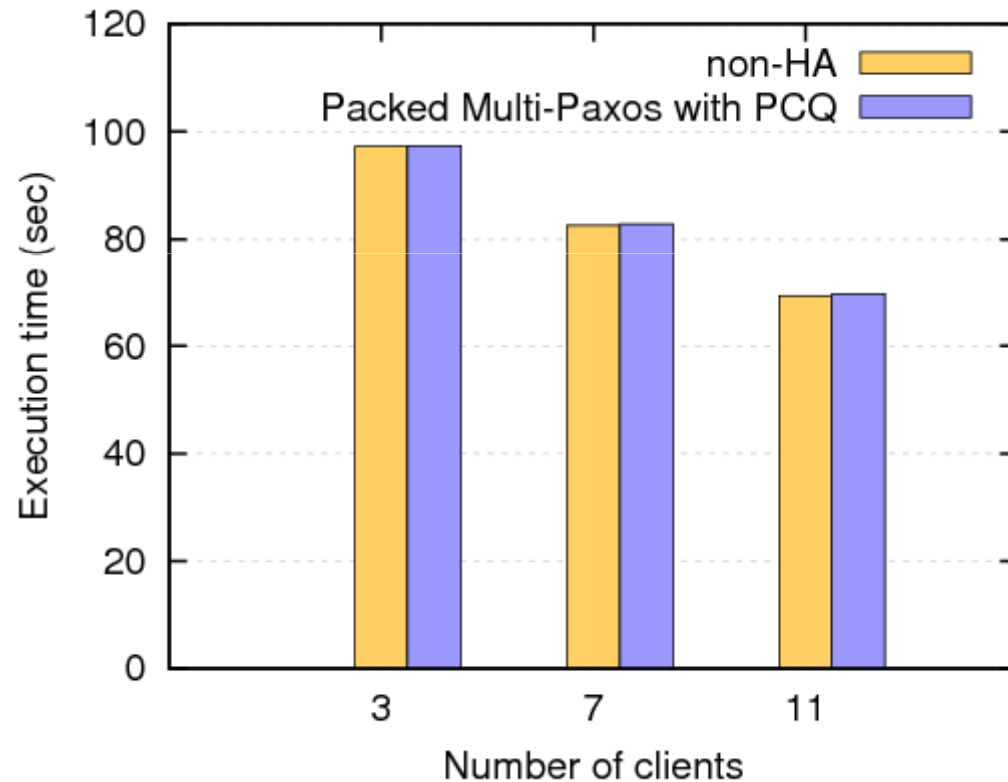
- ▶ Benchmark: *mdtest*
- ▶ Reported performance: file stat



- Configuration:
 - 12 clients, 1 OSD
 - Non-HA: 1 MDS
 - Others: 3 MDS
- *mdtest* parameters
 - 100 threads
 - 100,000 files

mpiBLAST

▶ Time spent in gene query and matching



- Configuration:

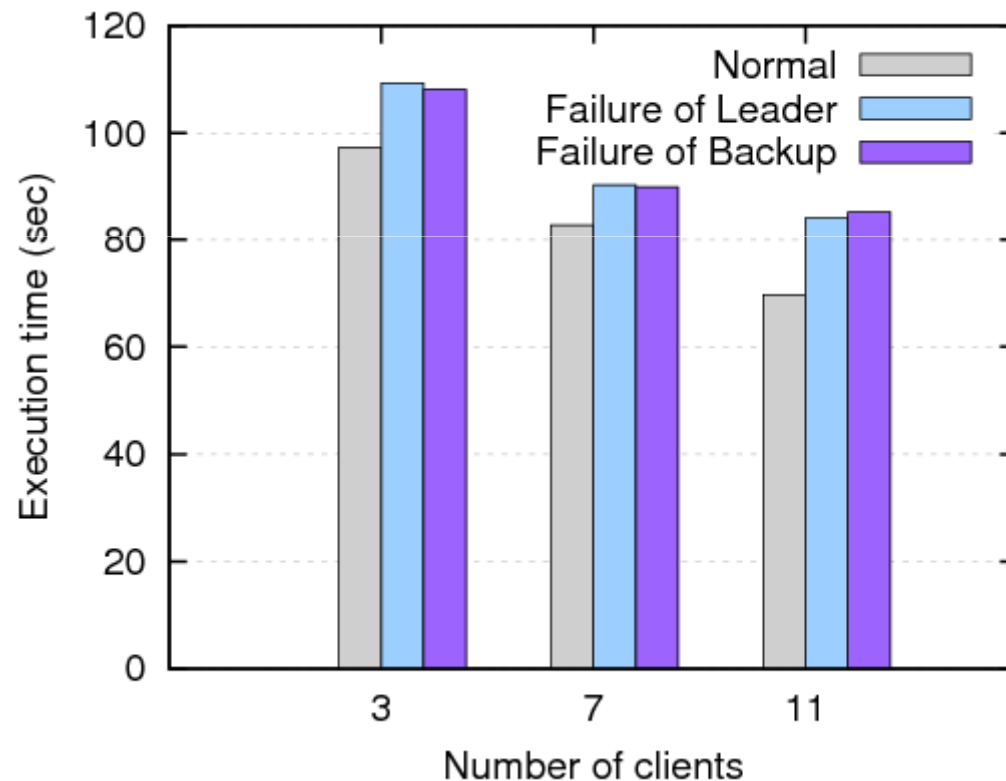
- 11 clients, 2 OSD
- Non-HA: 1 MDS
- Others: 3 MDS

- *mpiBlast* parameters

- Database size: 977MB

mpiBLAST

- ▶ Fault-tolerance
 - ▶ Inject failures during its execution



- Configuration:
 - 11 clients, 2 OSD
 - Non-HA: 1 MDS
 - Others: 3 MDS
- *mpiBlast* parameters
 - Database size: 977MB

Summary

- ▶ Our contributions
 - ▶ Adopt replication and the Paxos protocol to construct a highly available metadata management
 - ▶ To improve metadata throughput, propose a method that packs multiple log records into one message and controls execution order by a Coordination Queue

**Thank you!
&
Questions?**

Jin Xiong: xj@ncic.ac.cn